(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
31 May 2001 (31.05.2001)

PCT

(10) International Publication Number
WO 01/38970 A2

(51) International Patent Classification?: G06F 9/00

(21) International Application Number: PCT/US00/26669

(22) International Filing Date:
28 September 2000 (28.09.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/447,081 22 November 1999 (22.11.1999) US

(71) Applicant: ERICSSON INC. [US/US]; 7001 Development Drive, Research Triangle Park, NC 27709 (US).

(72) Inventor: DENT, Paul, W.; 637 Eaglepoint Road, Pittsboro, NC 27312 (US).

(74) Agent: MYERS BIGEL SIBLEY SAJOVEC, P.A.; P.O. Box 37428, Raleigh, NC 27627 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *Without international search report and to be republished upon receipt of that report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: BUFFER MEMORIES, METHODS AND SYSTEMS FOR BUFFERING HAVING SEPERATE BUFFER MEMORIES FOR EACH OF A PLURALITY OF TASKS

(57) Abstract: Methods, systems and buffer memories are provided that include one buffer per task rather than a single buffer, (the contents of which may be discarded or overwritten when switching tasks in a multi-tasking environment). The rate of cache misses is, therefore, dependent solely on a single task. Furthermore, the present invention provides systems and methods for transferring execution of instructions from one task to another task which may provide reduced overhead and improved utilization of processor cycles concurrently with access operations to a slower ROM to update a buffer memory responsive to a cache miss. The buffer memories are coupled to the ROM by a first, preferably wider, bus and to the processor by a second bus allowing buffer updates to one of the task's buffers while another task is executed from its buffer by the processor. An activity register is coupled to a task selection circuit to identify the highest priority task and select the corresponding buffer as the source of instructions for execution by the processor when the activity state of the various tasks in the multi-tasking environment changes. The activity state of a task is set responsive to the need for instruction execution for the particular task and also by the status of the associated buffer so that a task is set to inactive while buffer updates are performed, thereby allowing a different task to be designated as the highest priority task and executed from its own associated buffer during the buffer update.

WO 01/38970 A2

BUFFER MEMORIES, METHODS AND SYSTEMS FOR BUFFERING HAVING SEPARATE BUFFER MEMORIES FOR EACH OF A PLURALITY OF TASKS

BACKGROUND OF THE INVENTION

The present invention relates to buffer memories and, more particularly, to buffer memories for program instructions.

Buffer or cache memories are commonly used to allow a computer (processor) to execute program instructions faster than they could be directly read from a larger memory device containing the instructions of the program. This approach has also been applied to multi-tasking/multi-thread processors by providing a random access memory for holding part of each of a number of different quasi-simultaneously executing programs (tasks). The random access memory is typically divided into portions. The remainder of the program instructions are generally stored on a slower access speed memory device, such as a magnetic disk, and instructions are brought into the random access memory as required by overwriting that part of the program previously occupying the random access memory. As devices such as disk drives typically transfer data and instructions to memory slower than the processor can execute instructions, a partition which is waiting for new instructions from disk may be temporarily suspended and the processor instruction execution cycles may be re-allocated to a partition that already has its instructions and data in memory. However, the reallocation overhead load on the processor is generally significant although typically less than disk drive access times.

Complex memory management schemes are generally used in these prior art approaches to make the division of a program between various memory device locations invisible to the program user. Furthermore, in these prior art approaches the average processing speed impact caused by the memory management was typically reduced for the sake of permitting many users or programs to share the same computer which, for large expensive mainframe computers, provided a higher utilization of the processor capacity and associated capital investment. In light of the

relatively slow access speed of disk storage, the overhead in switching execution from one program to another was generally not significant.

Another approach to better matching processor speed to memory access speeds is to utilize both a faster access memory such as a random access memory and a slower access memory, such as a read only memory (ROM). One way of improving performance with such an architecture is to provide a larger bus width (preferably several instruction field widths) between the random access memory and the ROM than between the random access memory and the processor. This approach may allow multiple instructions to be loaded from the ROM into the random access memory for each ROM access operation, thereby increasing the effective access rate for the ROM. Such an approach may be visualized as a funnel having a wide mouth operating at a slow rate of flow (ROM to random access memory) and a narrow spout operating at a faster rate of flow (random access memory to processor). However, while this may be helpful where instructions are executed sequentially, where branch instructions, subroutine calls or returns and the like which jump execution to an instruction outside the group of instructions contained in the random access memory occur, the advantage of the larger bus width to the ROM may be, at least partially, lost and the processor required to wait. Frequent task changes in a multi-tasking environment may have a similar affect. Furthermore, the overhead of the prior art approaches to reallocating processor cycles to another task may exceed the time required to access the ROM which, while slower than accessing the random access memory, is still generally much faster than magnetic disk access operations.

It is further known to use the random access memory as a buffer (cache) in which to receive instructions from the slower ROM with the instructions being executed by the processor from the random access memory, including reading the same instructions a plurality of times in connection with execution of loop operations typically found in programs. The multiple executions of various instructions allows improved performance even though the rate of reading new, unexecuted instructions into the random access memory is still limited by the rate of access to the slower ROM.

Various different ways of implementing such a cache are known. One method is to split the program memory access field into a most significant part (MSP) and a least significant part (LSP). The cache memory is sized to store a number of words (comprising program instructions or data) that can be addressed by the LSP which

correspond to the contents of a ROM LSP address range having a particular MSP address and from which the words in the random access memory were fetched. When the processor generates an address, the LSP of the address addresses the cache to fetch an instruction (or data) plus the associated MSP. If the associated MSP matches the MSP generated by the processor, the instruction (or data) is confirmed as valid, *i.e.* deemed a cache hit, and is executed by the processor. Otherwise, a cache miss occurs and the address is routed to the larger, slower ROM instead, thereby causing the processor to wait to receive the correct instruction or data from the ROM. The correct instruction and its associated MSP then overwrite the original contents in the cache on the assumption that is has a greater probability than the original contents of being needed again soon.

With this approach to a cache, where a single cache is shared by many programs, each program can independently cause a cache miss and, furthermore, instructions (or data) fetched by one program (task or context) from the ROM can overwrite instructions belonging to another task so that the instructions associated with the other task are no longer resident when the other task attempts to resume execution. Accordingly, there is an increased likelihood that task switching (re-allocating processor cycles between different quasi-simultaneously executing programs) can cause an increased rate of cache misses.

These various approaches to buffering for program instructions all generally are less efficient in a multi-tasking environment. As task switching events become more frequent, the discarding of buffer contents will also be expected to occur more often with the processor in turn being caused to wait more frequently for buffer memory to refill. Accordingly, more efficient buffer memories for multi-tasking environments would be desirable.

SUMMARY OF THE INVENTION

According to the present invention, methods, systems and buffer memories are provided that include one buffer per task rather than a single buffer (the contents of which may be discarded or overwritten when switching tasks in a multi-tasking environment). The rate of cache misses is, therefore, dependent solely on a single task. Furthermore, the present invention provides systems and methods for transferring execution of instructions from one task to another task which may provide reduced overhead and improved utilization of processor cycles concurrently

with access operations to a slower ROM to update a buffer memory responsive to a cache miss. The buffer memories are coupled to the ROM by a first, preferably wider, bus and to the processor by a second bus allowing buffer updates to one of the task's buffers while another task is executed from its buffer by the processor. An activity register is coupled to a task selection circuit to identify the highest priority task and select the corresponding buffer as the source of instructions for execution by the processor when the activity state of the various tasks in the multi-tasking environment changes. The activity state of a task is set responsive to the need for instruction execution for the particular task and also by the status of the associated buffer so that a task is set to inactive while buffer updates are performed, thereby allowing a different task to be designated as the highest priority task and executed from its own associated buffer during the buffer update.

In one embodiment of the present invention, a buffer memory for a multi-tasking processor having an associated first memory that contains program code instructions associated with a plurality of tasks is provided. The buffer memory includes a buffer memory circuit coupled to the first memory over a first data bus and to the processor over a second data bus. The buffer memory circuit includes a first buffer memory associated with a first one of the plurality of tasks having an associated size smaller than a size of the first memory and a second buffer memory associated with a second one of the plurality of tasks having an associated size smaller than the size of the first memory. One of the first buffer memory and the second buffer memory is read accessible while the other is being write accessed. The buffer memory also includes a task selection circuit that selects one of the first buffer memory or the second buffer memory as an active memory based on which of the first one of the plurality of tasks and the second one of the plurality of tasks is active.

In a further embodiment, the buffer memory further includes a controller coupled to the task selection circuit that updates the first buffer memory and the second buffer memory from the first memory and sets the first one of the plurality of tasks to an inactive state while the first buffer memory is updated and the second one of the plurality of tasks to an inactive state while the second buffer memory is updated. In addition, the buffer memory circuit in one embodiment includes a first set of registers associated with the first one of the plurality of tasks and a second set of registers associated with the second one of the plurality of tasks. The controller may execute the first one of the plurality of tasks and the second one of the plurality of

tasks. The first set of registers may be used by the controller when executing the first one of the plurality of tasks and the second set of registers may be used by the controller when executing the second one of the plurality of tasks. The first set of registers and the second set of registers may each include a program counter register containing the address of a next instruction to be executed for the associated task. The program counter registers may include a lower bit portion associated with a location in the associated one of the first buffer memory or the second buffer memory and an upper bit portion associated with a location in the first memory.

In another embodiment of the present invention, the buffer memory further includes a first-in first-out (FIFO) memory that queues the upper bit portion associated with a location in the first memory to address the first memory for access to the first memory. The FIFO memory may be configured to output a next queued upper bit portion to the first memory responsive to the controller. In one embodiment, the task selection circuit further includes an activity register having a first flag associated with the first one of the plurality of tasks and a second flag associated with the second one of the plurality of tasks, the first flag and the second flag being set to indicate an activity status of the associated task. The first data bus is preferably wider than the second data bus.

In another aspect of the present invention, systems and methods are provided for program buffering in a multi-tasking environment having a plurality of tasks for execution. A plurality of buffer memories are provided, respective ones of which are associated with each of the plurality of tasks. A portion of an instruction set of the associated one of the plurality of tasks is stored in each of the respective ones of the plurality of buffer memories. A highest priority one of the plurality of tasks is determined and the portion of an instruction set associated with the highest priority one of the plurality of tasks is executed until a next instruction for the highest priority one of the plurality of tasks is not in the associated one of the plurality of buffer memories. An update of the associated one of the plurality of buffer memories with a second portion of the instruction set of the highest priority one of the plurality of tasks is obtained when the next instruction for the highest priority one of the plurality of tasks is not in the associated one of the plurality of buffer memories. The portion of an instruction set associated with a next priority one of the plurality of tasks is executed while obtaining an update of the associated one of the plurality of buffer memories with a second portion of the instruction set of the highest priority one of the

plurality of tasks.

In another embodiment of the present invention, activity indicators associated with the tasks have an active state indicating readiness to execute instructions and an inactive state indicating that the associated one of the plurality of tasks has at least one of: no instructions requiring execution, or: an associated one of the plurality of buffer memories is obtaining an update. The highest priority one of the plurality of tasks is selected from ones of the plurality of tasks having an associated activity indicator in an active state. In one embodiment, the activity indicator for the highest priority one of the plurality of tasks is set to an inactive state while obtaining an update. The update may be obtained by placing an address associated with a location in an instruction memory containing the second portion of the instruction set in a FIFO register. The address placed in the FIFO register is retreived from the FIFO register to obtain the update when a wait indication indicates that the instruction memory is available for read access. The wait indication is set while obtaining the update to indicate that the instruction memory is not available and then reset after obtaining the update.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a schematic block diagram illustrating a processor interfaced to a buffer memory according to an embodiment of the present invention;

FIG. 2 is a schematic block diagram further illustrating the buffer memory of **FIG. 1**;

FIG. 3 is a schematic block diagram of a processor having multiple register banks for use in a multi-tasking environment;

FIG. 4 is a flowchart illustrating operations for program buffering in a multi-tasking environment according to an embodiment of the present invention;

FIG. 5 is a flowchart illustrating operations related to initiation of a buffer update according to an embodiment of the present invention; and

FIG. 6 is a flowchart illustrating operations related updating task priority encoding for an embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

The present invention now will be described more fully hereinafter with reference to the accompanying drawings, in which preferred embodiments of the

invention are shown. This invention may, however, be embodied in many different forms and should not be construed as limited to the embodiments set forth herein; rather, these embodiments are provided so that this disclosure will be thorough and complete, and will fully convey the scope of the invention to those skilled in the art. As will be appreciated by those of skill in the art, the present invention may be embodied as methods or devices. Accordingly, the present invention may take the form of a hardware embodiment, a software embodiment or an embodiment combining software and hardware aspects.

The present invention will now be further described with reference to the block diagram illustration of **FIG. 1**. As shown in **FIG. 1**, a buffer memory **10** according to an embodiment of the present invention is coupled to an associated memory such as a ROM **20** that contains program code instructions associated with a plurality of tasks. Both the buffer memory **10** and the ROM **20** are coupled to a multitasking processor **30**. The buffer memory **10** adapts the larger slower memory or ROM **20** to operate with the processor **30**. As further shown in the illustrated embodiment, the ROM **20** is coupled to the buffer memory **10** over a bus **35** having a relatively large bus width, such as 256 bits (32 bytes). The buffer memory **10** is coupled to the processor **30** over a relatively smaller width bus including a byte address portion **40**, such as a 5 bit byte address representing the least significant bit (LSB) portion of an address and a byte read data bus portion **45**.

The processor **30** in the illustrated embodiment is shown as a byte-oriented processor, although fast, modern processors often use larger word sizes, such as 16, 32 or even 64 bits, and such processors may also beneficially be used according to the teachings of the present invention. Such modern processors, like the Pentium™ line from Intel Corporation, are typically optimized for speed rather than memory utilization, but other applications, such as wireless phones and data devices (such as the Palm Pilot™ line available from 3Com Corporation, which are battery operated, typically have other constraints on memory size and power consumption. Accordingly, better byte-oriented processor architectures may provide better memory and power utilization than 32 bit or 64 bit processors for such applications. An example of such a byte oriented processor suitable for use for the present invention is described in United States Patent Application No. 09/264,795 entitled "Efficient Microprocessor Architecture" (attorney docket number P10380) which is incorporated by reference herein.

Regardless of the width of the processor bus **40, 45**, it is generally the case that the processor **30** can execute an instruction in a shorter time than one read operation from the ROM **20**. A typical type of ROM **20** used in battery-portable type applications is flash ROM, having an access time typically of approximately 50 nanoseconds (ns). In contrast, a typical processor execution cycle currently may well be an order of magnitude shorter in time, such as 5 ns. Accordingly, pursuant to the present invention, the ROM **20** is configured to deliver multiple processor instructions on each read operation through the use of the wide bus **35**. As shown in **FIG. 2**, the bus **35** from the ROM **20** to the buffer memory **10** has a bus width of 32 bytes as compared with the one byte width of the data bus **45** coupling the buffer memory 10 to the processor **30**. Thus, assuming the processor **30** executes its instructions linearly, that is with no branches, the processor **30** will, using the above exemplary timings, utilize 160 ns (32 x 5 ns) to execute the 32 bytes of instruction stored in an associated buffer memory **10** utilizing a 32 byte buffer cache size. During this time, the ROM **20** is provided sufficient time to retrieve the next 32 bytes and place them on the wider bus **35** in anticipation of updating the contents of the buffer memory **10**.

Additional control lines are further illustrated in **FIG. 1** for use in controlling memory operations according to the present invention. The processor **30** is coupled to the ROM **20** by a wait status line **65** which the processor **30** can check when it has finished executing instructions from the buffer memory **10** to determine if it is still waiting for the ROM **20** to place the next 32 bytes on the bus **35**. The replenish line **50** connects the processor **30** to the buffer memory **10.** When the wait status line **65** is reset, indicating that the data is placed on the bus **35**, the replenish line **50** may be used by the processor **30** to cause the buffer memory **10** to load the data from the bus **35** into the buffer memory **10** overwriting the original data. Note that this overwriting alternatively may take place progressively, one byte at a time, ahead of the processor **30** needing a new byte. This approach requires attention to potential branch backs by the processor **30** which would appear as a miss if the previous byte had already been overwritten. Accordingly, to optimize performance, the likelihood of jumps or branches of various distances should be analyzed and utilized in selecting an appropriate cache buffer size that would encompass the most frequent jump distances, thus allowing loop execution without memory accesses to the ROM **20**.

As also shown in **FIG. 1** the upper or most significant portion (MSP) of the address provided on the address MSP bus **55** from the processor **30** is coupled to the

ROM **20** along with a read line **60**. During linear execution, after reading data off of the bus **35** into the buffer memory **10**, the processor **30** preferably anticipates continued linear execution and increments the address on the address MSP bus **55** to the ROM **20** in order to prefetch the next 32 bytes so that they may be waiting on the bus 35 before the buffer memory **10** exhausts available memory. The read line **60** is used to initiate a read responsive to an address on the address MSP bus 55.

It should also be understood that, when a nonlinear code execution sequence occurs, such as a branch, the processor **30** will change the address MSP bus **55** value to a different nonsequential value in order to read instructions from an appropriate location in the ROM **20**. This will be expected to cause the ROM 20 to set the wait status line **65** to the processor **30** while it fetches the new 32 byte block of data. (instructions). As will be described further herein, the present invention provides useful utilization of the processor **30** during such wait periods.

As shown in the illustrated embodiment of **FIG. 2**, the buffer memory **10** includes a buffer memory circuit **200** and a task selection circuit **205**. The buffer memory circuit **200** includes a plurality of caches **210a - 210d** each of which is associated with one of the plurality of tasks (contexts) supported by the processor **30**. Each of the respective buffer memories **210a – 210d** has a size smaller than the ROM **20**, for example a 32 byte cache size may be utilized for the caches **210a – 210d**. Each of the caches (buffer memories) **210a – 210d** is coupled to the ROM **20** over the wide bus **35** and is further connected to the processor **30** over the narrower data bus **45** and an address LSP bus **40**. Furthermore, each of the buffer memories **210a – 210d** is read accessible while others of the buffer memories **210a – 210d** are being write accessed.

As shown in the embodiment of **FIG. 2** the buffer memory circuit **200** includes register banks **215a – 215d** which are separately accessible to the processor **30** over the register bank bus **220**. Accordingly, the buffer memory circuit **200** provides both a per context buffer memory **210a-210d** and a per context register bank **215a – 215d**. The buffer memory circuit **200** receives an address from the task selection circuit **205** which selects the buffer memory **210a – 210d** and the register bank 215a-**215d** to be used currently by the processor **30**, thereby establishing a current executing task. In the illustrated embodiment, the task or context selection address is an 8 bit address provided over bus **240** from the task selection circuit 205 to

the buffer memory circuit **200** and further fed back as an input to the task selection circuit **205** indicating a current context (or task).

Preferably, one register in each of the respective register banks **215a – 215d** is a program counter which holds the address of the next instruction to be executed by the respective task. The least significant part of the instruction address from the program counter may then be output by the processor **30** on the bus **40** to become the address for retrieving the next instruction from the selected buffer memory **210a – 210d**. The most significant part of the address may be output by the processor 30 to the ROM **20** when the currently executing task exhausts the supply of instructions from its associated buffer memory **210a – 210d**. As described previously, during memory updates, the ROM **20** sets the wait status line **65** to the processor **30** which in turn generates updated task activity information for use by the task selection circuit **205** as will be described further herein. The task activity information is used by the task selection circuit **205** in selecting one of the buffer memories **210a-210d** of the buffer memory circuit **200** as an active memory based on which of the tasks is selected to be active. The task selection circuit **205** includes a bit addressing circuit **225**, an activity register **230** and a priority encoding circuit **235**.

Operations for updating task activity status in connection with buffer memory update operations will now be further described with reference to the flow chart illustration of **FIG. 5**. When the instructions in one of the buffer memories **210a – 210d** for an active task is exhausted, the processor **30** checks to determine if the wait status line **65** is activated (set) or inactivated (reset) (block **500**). If the wait status line **65** is not activated, the ROM **20** is available for memory reads and a new address is provided on the address MSP bus **55** to ROM **20** to initiate the update of the buffer memory **210a-210d** (block **505**). In turn, the ROM **20** generates a wait signal (*i.e.* activities or sets) on the wait status line **65** to the processor **30** which, in turn, issues a set to inactive signal to the bit addressing circuit **225** of the task selection circuit **205** (block **510**). This change to inactive status is further entered into the activity register **230** by the bit addressing circuit **225**. More particularly, the bit addressing circuit **225**, upon receipt of the set context inactive signal from the processor **30**, transfers the current context selection input from the priority encoding circuit **235** through to the activity register **230** on the illustrated 8 bit bus for the embodiment of **FIG. 2** (block **515**). The bit addressing circuit **225** further sends a reset signal to the activity register **230** so that the addressed bit location will be set to inactive (block **520**). As the bits

of the activity register **230** (256 bits associated with 256 different buffer memories **210a-210d** in the illustrated embodiment) are preferably hardwired from the activity register **230** to the priority encoding circuit **235**, this update will promptly cause the priority encoding circuit **235** to determine the next highest priority active task and output the appropriate task selection for the new highest priority task on the output bus **240** (block **525**). This in turn selects a new buffer memory **210a – 210d** and register bank **215a-215d** to be utilized by the processor **30**.

The selected new context (task) was, presumptively, in an active state. Otherwise, it would not be selected by the priority encoding circuit **235** (as if it was, the task would have been flagged as inactive and, therefore, not selected). Therefore, the associated buffer memory **210a-210d** is not exhausted and no new MSP address part need be output by the processor **30** to the ROM **20** over the bus **55**. Furthermore, such an address update will not be allowed so long as a wait signal on the wait status line **65** from the ROM **20** continues to be asserted. Should a buffer memory **210a-210d** become exhausted before the ROM **20** wait state is reset, a next most significant part address may be placed into an address request queue such as a First In First Out (FIFO) register where it may be queued until the previous memory wait signal disappears (block **530**). In addition, queuing in the FIFO register provides an indication that the task cannot receive its new buffer memory refill immediately and operations related to inactivating a task as described above may be executed for the waiting task. A new task will thereby be selected and operations continue as described above. The alternative embodiment using queuing is illustrated by the FIFO register **245** contained in the processor **30** illustrated in **FIG. 2**.

As will be appreciated by those of skill in this art, the above-described aspects of the present invention in **FIGs. 1** and **2** may be provided by hardware, software, or a combination of the above. For example, the task selection circuit **205** may be implemented in part as code executing on a processor although it is preferred that the priority encoded circuit be a hardwired circuit.

As noted above, the present invention is directed towards buffer memories for processors capable of multi-tasking, i.e., of executing stored instructions for a plurality of different tasks, with the respective tasks being invoked in a prioritized order when they have work to perform as indicated by a status or activity flag. The status flags may be set (active) or reset (inactive) by the processor detecting an interrupt informing the processor of an external event. Patent application serial

number 09/338,732 entitled " Improved Context Switch and Rescheduling of a Processor" which is incorporated herein by reference, describes a processor having special hardware and a plurality of registers each associated with one of a plurality of tasks such that the processor can more rapidly detect a high priority task changing state from inactive to active and switch processor resources from a lower priority task.

Such a processor is further illustrated in **FIG. 3** herein. As shown in **FIG . 3**, the processor architecture may reduce the time for switching from one task to another to, theoretically, one clock cycle. This is provided in part by a separately selectable set of registers **300** for each task, thereby potentially avoiding the overhead of saving register values from one task in a stack and retrieving register values for another task from another stack upon task switching. In addition, rapid task switching may be facilitated by use of the hardware register **305** containing indications of task activity or inactivity, with a hardware logic circuit **310** providing a translation from the contents of the hardware register **305** to an address for the one of the set of registers **300** associated with the highest priority, active task. Accordingly, the processor architecture of **FIG. 3** may be able to switch from one task to another in the time typically taken by the processor to execute a single instruction.

Operations of the present invention will now be described with respect to the flowchart illustrations of **FIGs. 4** to **6**. It will be understood that each block of the flowchart illustrations and the block diagram illustrations, and combinations of blocks in the flowchart and block diagram illustrations can be implemented by computer program instructions or hardwired sequential logic. These program instructions may be provided to a processor to produce a machine, such that the instructions which execute on the processor create means for implementing the functions specified in the flowchart and block diagram block or blocks. The computer program instructions may be executed by a processor to cause a series of operational steps to be performed by the processor to produce a computer implemented process such that the instructions which execute on the processor provide steps for implementing the functions specified in the flowchart and block diagram block or blocks.

Accordingly, blocks of the flowchart illustrations and the block diagrams support combinations of means for performing the specified functions, combinations of steps for performing the specified functions and program instruction means for performing the specified functions. It will also be understood that each block of the flowchart illustrations and block diagrams, and combinations of blocks in the

flowchart illustrations and block diagrams, can be implemented by special purpose hardware-based systems which perform the specified functions or steps, or combinations of special purpose hardware and computer instructions.

As shown in FIG. 4, operations begin at block 400 when the buffer memories 210a-210d are associated with each of a plurality of tasks supported by the processor 30 in a multi-tasking environment. This may occur upon switching on the power to the processor, typically causing a power-up reset. The need for power-up reset is well known in the art and is not shown in the figures, although it may be provided in embodiments of the present invention.

Upon power-up, it can generally be assumed that all registers and buffer memories are empty, and the power-up reset would typically comprise setting any indicators that the buffer memories were empty. In prior art processors, power-up reset was typically arranged to cause a power-up interrupt and thus a branch to a specific location in the program memory where the first instruction to be executed would be found. In the prior art it was also common to distinguish between INTERRUPT level operation and NON-INTERRUPT or BASE level operation. Execution of instructions at any interrupt level would generally always take priority over execution at base level.

In a preferred embodiment of the present invention, it is not necessary to distinguish between base level and interrupt level tasks, but merely to reserve the highest priority tasks to be those called "INTERRUPT LEVEL" tasks in the prior art. Accordingly, the highest priority task in the present invention may be allocated to power-up reset. Power-up reset may be hardwired to cause the contents of the highest priority buffer memory to be fetched from a predetermined address in the slower ROM, for example, from the 256-bit ROM area beginning at address zero. Power-up reset may also set the program counter to the value which could cause execution of the first instruction of this initial buffer fill, for example, the first byte or word of the buffer. The activity flag of the power-up reset task would then preferably also automatically be set to active upon detecting power-up, and the activity flags of other tasks set to inactive. When the first 256-bits of the power-up code had been executed, exhausting the buffer, the already-described mechanism for replenishing the buffer would operate, except that, with no other active tasks to occupy the ROM latency time, the lowest priority "power down" or "sleep mode" operation may be selected, effectively turning off the processor such as by stopping or slowing down its clock

and saving power while the next buffer fill is being loaded from ROM. Upon disappearance of the ROM wait-state indicating that the next buffer fill is available, the power-up reset task would resume execution.

Thus, the power-up task can comprise a program of any length which will commence execution automatically upon switching on the power. This program may initialize the program counters of all of the other, lower-priority tasks that are desired to be resident at switch on, using an initialization table programmed into the ROM. Some of these tasks may be other "interrupt-level" tasks of lower priority than the power-up task, which, although they may be set to active, will preferably not be executed until the power-up task becomes inactive upon completing power-up initialization. The power-up program may remain resident after initialization, ready to resume execution at a different part of its code for handling emergency power-down functions.

External power management circuitry can be arranged to detect imminent loss of the power to the processor and cause a power-down interrupt. The power-down interrupt can cause resumption of execution of the highest priority task which would now preferably execute a sequence of instructions to place the processor in a state to accept loss of power gracefully. For example, there can be enough time left, if the power-loss warning is properly arranged, to save the entire contents of the register set **300** in non-volatile EEPROM memory, or to save variables from RAM to EEPROM that should not be lost.

After initialization is complete, which may include setting an initial task to the active state, the initially active task may commence execution, or alternatively, one of the initially inactive tasks associated with a particular hardware interrupt may become set to active through interrupt detection circuitry detecting an interrupt from the particular hardware, such as a clock-timer or keyboard scanner. Depending upon how the programmer has determined that the software shall respond to such external stimuli, an initially active task may then spawn other tasks by giving another register set an initial program counter value to fetch a first buffer fill from ROM, and setting the task to the active state. The ROM may contain re-entrant program routines that are not part of any particular priority level, but which may be called into operation by a program executing at any priority level for performing commonly desired functions such as spawning another task, killing a task and freeing upon the associated priority level, setting a task to active or inactive, or passing messages between tasks in a

standard format variously called "semaphores" or "Device Control Blocks" in the prior art. These commonly used routines may constitute an operating system such as WINDOWS™ available from Microsoft Corporation, LINUX or EPOC, to name but a few currently available operating systems. Because of the novelty of the processor architecture described in this invention, operating systems could advantageously be written specifically for the inventive processor, in order to use its capabilities to execute code fast and power-efficiently to maximum benefit in a given application, such as a cellular phone or other portable battery operated device.

Referring again to **FIG. 4**, a portion of an instruction set of each of the associated ones of the plurality of tasks are stored in each of the respective ones of the buffer memories **210a – 210d** (block **405**). A highest priority one of the tasks for execution is identified (block **410**). The portion of the instruction set associated with the identified highest priority one of the tasks is then executed (block **415**) until a next instruction for the highest priority one of the plurality of tasks is not contained in its associated buffer memory **210a – 210d** thereby requiring an update (block **420**).

An update of the respective buffer memory **210a – 210d** with a second portion of the instruction set for the associated task is then initiated when the next instruction for that task is not in the associated buffer memory **210a – 210d** (block **425**). During the wait period while the update is obtained, a new task, or next priority one of the plurality of tasks, is selected for execution by the processor **30** (block **430**). The portion of the instruction set for the new task contained in its respective one of the buffer memories **210a – 210d** is then executed while obtaining an update of the buffer memory contents for the previous high priority task (block **435**). The update of the buffer memory for one task may occur concurrently with execution of instructions for the next selected task as the buffer memories **210a – 210d** may each be accessed for read operations by the processor **30** while the write operations to the particular buffer memory **210a – 210d** being updated are occurring concurrently. Operations continue each time an activity status for one of the tasks changes (block **440**) initiating a re-encoding of the task selection to determine if a different task should be selected for execution (block **410**).

The associated task priority selection operations described above with reference to block **410** and block **430** are preferably supported through the use of activity indicators for each of the tasks as described previously with reference to **FIG. 2** where the activity indicators have an active state indicating a readiness to execute

instructions and an inactive state indicating the associated task either has no instructions requiring execution or has an associated buffer memory **210a - 210d** that is in the process of or requires an update. Accordingly, the highest priority one of the plurality of tasks at any given update is selected from those tasks having an associated activity indicator in an active state. Furthermore, as described previously with reference to **FIG. 5**, when a buffer memory update is initiated for a particular task, the activity indicator for that task is preferably set to an inactive state while obtaining the update so that a different task will be selected for execution.

Operations at block **425** in obtaining an update when utilizing the illustrated embodiment of **FIG. 2** including a FIFO register **245** begin by placing an address associated with a location in the ROM (or instruction memory) **20** containing the next portion of the associated instruction set in the FIFO register **245**. The address is then retrieved in sequence from the FIFO register **245** to obtain the update when the wait status line **65** indicates that the ROM memory **20** is available for read access. The wait status line **65** is then set while obtaining the update to indicate that the ROM **20** is not available and then reset after obtaining the update to indicate that the ROM **20** is again available. The FIFO register **245** thus supports sequentially processing updates to the respective buffer memories **210a – 210d** while setting to inactive any tasks having an address in the FIFO register **245** indicating a need for an update to its corresponding buffer memory **210a – 210b**. Accordingly, all tasks awaiting a memory update through the FIFO register **245** are set to an inactive state while awaiting completion of the update so that another one of the plurality of tasks having instructions available in its respective buffer memory **210a – 210d** may execute while buffer memory update operations occur.

Upon completion of a buffer memory update for each respective task, its corresponding activity indicator is again set to indicate an active state thereby initiating an update so that, where there is a higher priority task than the currently executing task, the task selection circuit **205** may return operations to the higher priority task which now has a refreshed buffer memory.

Operations related to resetting a task to an active state following completion of a buffer memory update will now be further described for an embodiment of the present invention with reference to the flow chart illustration of **FIG. 6**. Operations begin at block **600** when the wait state ends responsive to completion of a read operation to place a new block of instructions in the respective buffer memory 210a –

210d. On removal of the wait signal, the processor 30 determines which task had been placed in the temporarily inactive state due to the wait state which has just been removed (block 605). This may be provided where nested waits exist in the FIFO register 245 by storing in the FIFO register 245 a task number associated with the MSP address input to the FIFO register 245. Accordingly, when the ROM 20 resets the wait signal on the wait status line 65, the task number of the waiting task stored in the FIFO register 245 can be transferred (as illustrated by the doted line path 250 in FIG. 2) to the bit addressing circuit 225 (block 610) so that the correct activity flag can be set back to the active state in the activity register 230 (block 615). The task selection output may then be updated by the priority encoding circuit 235 in light of the changed activity status state of the task which just completed its buffer memory update (block 620).

The bus connection 40, 45 between the processor 30 and the task selection circuit 205 further allows the processor 30 to set the activity flags associated with any task to active or inactive under program control. For example, program control may be provided to determine whether a particular task has any activities currently pending requiring execution. In one embodiment, this dual activity flag control may be provided through the use of activity flags which have 2 bits associated with the different respective causes of inactivity (*i.e.*, due to memory wait conditions or to a deliberately programmed inactivity condition). The separate activity bits indicating the separate causes for suspending execution of a particular task may in turn be input to an AND gate, the output of which is provided to the priority encoding circuit 235 so that a task is only selected as the current task to execute when it is neither waiting for a memory update operation nor has suspended operations under control of another program.

Accordingly, when practicing the present invention, a processor may be allowed to execute other, lower priority tasks that have still not exhausted program caches while waiting for the program cache of a higher priority task to refill from a slower memory. In addition to providing for rapid task switching, the present invention provides a potential for lower priority tasks to receive a plurality of instruction execution cycles during each access of a slower memory by higher priority tasks, thereby beneficially utilizing a greater proportion of the capacity of a high speed processor. As an example, with reference to the illustrated embodiment of FIG. 2, up to eight instruction execution cycles may be available for a lower priority

task during memory accesses by a higher priority task assuming that an access to the ROM **20** requires ten execution cycles while only one cycle is used for task switching out of the current task which is waiting for the memory **20** and one cycle is used for switching back to that task again when the memory **20** has completed its memory update operations.

In the drawings and specification, there have been disclosed typical preferred embodiments of the invention and, although specific terms are employed, they are used in a generic and descriptive sense only and not for purposes of limitation, the scope of the invention being set forth in the following claims.

THAT WHICH IS CLAIMED IS:

1. A buffer memory for a multi-tasking processor having an associated first memory that contains program code instructions associated with a plurality of tasks, the buffer memory comprising:

a buffer memory circuit coupled to the first memory over a first data bus and to the processor over a second data bus, the buffer memory circuit comprising:

a first buffer memory associated with a first one of the plurality of tasks having an associated size smaller than a size of the first memory; and

a second buffer memory associated with a second one of the plurality of tasks having an associated size smaller than the size of the first memory;

wherein one of the first buffer memory and the second buffer memory is read accessible while the other is being write accessed; and

a task selection circuit that selects one of the first buffer memory or the second buffer memory as an active memory based on which of the first one of the plurality of tasks and the second one of the plurality of tasks is active.

2. The buffer memory of Claim 1 further comprising a controller coupled to the task selection circuit that updates the first buffer memory and the second buffer memory from the first memory and sets the first one of the plurality of tasks to an inactive state while the first buffer memory is updated and the second one of the plurality of tasks to an inactive state while the second buffer memory is updated.

3. The buffer memory of Claim 2 wherein the buffer memory circuit further comprises a first set of registers associated with the first one of the plurality of tasks and a second set of registers associated with the second one of the plurality of tasks.

4. The buffer memory of Claim 3 wherein the controller executes the first one of the plurality of tasks and the second one of the plurality of tasks and wherein the first set of registers is used by the controller when executing the first one of the plurality of tasks and the second set of registers is used by the controller when executing the second one of the plurality of tasks.

5. The buffer memory of Claim 4 wherein the first set of registers and the second set of registers each include a program counter register containing the address of a next instruction to be executed for the associated task.

6. The buffer memory of Claim 5 wherein the program counter registers include a lower bit portion associated with a location in the associated one of the first buffer memory or the second buffer memory and an upper bit portion associated with a location in the first memory.

7. The buffer memory of Claim 6 further comprising a first-in first-out (FIFO) memory that queues the upper bit portion associated with a location in the first memory to address the first memory for access to the first memory.

8. The buffer memory of Claim 7 wherein the FIFO memory is configured to output a next queued upper bit portion to the first memory responsive to the controller.

9. The buffer memory of Claim 4 wherein the task selection circuit further comprises an activity register having a first flag associated with the first one of the plurality of tasks and a second flag associated with the second one of the plurality of tasks, the first flag and the second flag being set to indicate an activity status of the associated task.

10. The buffer memory of Claim 4 wherein the first data bus is wider than the second data bus.

11. A method for program buffering in a multi-tasking environment having a plurality of tasks for execution comprising the steps of:

providing a plurality of buffer memories;

associating respective ones of the plurality of buffer memories with each of the plurality of tasks;

storing in each of the respective ones of the plurality of buffer memories a portion of an instruction set of the associated one of the plurality of tasks;

determining a highest priority one of the plurality of tasks;

executing the portion of an instruction set associated with the highest priority one of the plurality of tasks until a next instruction for the highest priority one of the plurality of tasks is not in the associated one of the plurality of buffer memories;

obtaining an update of the associated one of the plurality of buffer memories with a second portion of the instruction set of the highest priority one of the plurality of tasks when the next instruction for the highest priority one of the plurality of tasks is not in the associated one of the plurality of buffer memories; and

executing the portion of an instruction set associated with a next priority one of the plurality of tasks while obtaining an update of the associated one of the plurality of buffer memories with a second portion of the instruction set of the highest priority one of the plurality of tasks.

12. The method of Claim 11 further comprising the step of providing an activity indicator for each of the plurality of tasks, the activity indicators having an active state indicating readiness to execute instructions and an inactive state indicating that the associated one of the plurality of tasks has at least one of no instructions requiring execution or has an associated one of the plurality of buffer memories which is obtaining an update.

13. The method of Claim 12 wherein the step of determining a highest priority one of the plurality of tasks further comprises the step of selecting the highest priority one of the plurality of tasks from ones of the plurality of tasks having an associated activity indicator in an active state and wherein the step of determining a next priority one of the plurality of tasks further comprises the step of selecting the next priority one of the plurality of tasks from ones of the plurality of tasks having an associated activity indicator in an active state.

14. The method of Claim 13 wherein the step of obtaining an update of the associated one of the plurality of buffer memories with a second portion of the instruction set of the highest priority one of the plurality of tasks when the next instruction for the highest priority one of the plurality of tasks is not in the associated one of the plurality of buffer memories further comprises the step of setting the activity indicator for the highest priority one of the plurality of tasks to an inactive state while obtaining an update.

15. The method of Claim 14 wherein the step of obtaining an update of the associated one of the plurality of buffer memories with a second portion of the instruction set of the highest priority one of the plurality of tasks when the next instruction for the highest priority one of the plurality of tasks is not in the associated one of the plurality of buffer memories further comprises the steps of:

placing an address associated with a location in an instruction memory containing the second portion of the instruction set in a FIFO register;

retrieving the address placed in the FIFO register from the FIFO register to obtain the update when a wait indication indicates that the instruction memory is available for read access;

setting the wait indication while obtaining the update to indicate that the instruction memory is not available; and then

resetting the wait indication after obtaining the update.

16. The method of Claim 15 wherein the step of retrieving the address placed in the FIFO register from the FIFO register to obtain the update when a wait indication indicates that the instruction memory is available for read access is preceded by the steps of:

retrieving from the FIFO register an address associated with obtaining an update for a one of the plurality of buffer memories other than the one of the plurality of buffer memories associated with the highest priority one of the plurality of tasks ahead of the address associated with a location in an instruction memory containing the second portion of the instruction set to obtain an update;

setting the wait indication while obtaining an update for a one of the plurality of buffer memories other than the one of the plurality of buffer memories associated with the highest priority one of the plurality of tasks; and then

resetting the wait indication after obtaining an update for a one of the plurality of buffer memories other than the one of the plurality of buffer memories associated with the highest priority one of the plurality of tasks.

17. The method of Claim 15 wherein the step of resetting the wait indication after obtaining the update further comprises the step of setting the activity indicator for the highest one of the plurality of tasks to an active state.

18. The method of Claim 17 wherein the step of setting the wait indication while obtaining the update to indicate that the instruction memory is not available further comprises the step of setting the activity indicator for the highest one of the plurality of tasks to an inactive state while obtaining the update.

19. The method of Claim 18 wherein the step of determining a highest priority one of the plurality of tasks further comprises the step of determining a highest priority one of the plurality of tasks task responsive to changes in the activity indicators.

20. A system for program buffering in a multi-tasking environment having a plurality of tasks for execution comprising:

means for providing a plurality of buffer memories;

means for associating respective ones of the plurality of buffer memories with each of the plurality of tasks;

means for storing in each of the respective ones of the plurality of buffer memories a portion of an instruction set of the associated one of the plurality of tasks;

means for determining a highest priority one of the plurality of tasks;

means for executing the portion of an instruction set associated with the highest priority one of the plurality of tasks until a next instruction for the highest priority one of the plurality of tasks is not in the associated one of the plurality of buffer memories;

means for obtaining an update of the associated one of the plurality of buffer memories with a second portion of the instruction set of the highest priority one of the plurality of tasks when the next instruction for the highest priority one of the plurality of tasks is not in the associated one of the plurality of buffer memories;

means for determining a next priority one of the plurality of tasks;

means for executing the portion of an instruction set associated with the next priority one of the plurality of tasks while obtaining an update of the associated one of the plurality of buffer memories with a second portion of the instruction set of the highest priority one of the plurality of tasks.

21. The system of Claim 20 further comprising an activity indicator for

each of the plurality of tasks, the activity indicators having an active state indicating readiness to execute instructions and an inactive state indicating that the associated one of the plurality of tasks has at least one of no instructions requiring execution or has an associated one of the plurality of buffer memories which is obtaining an update.

22. The system of Claim 21 wherein the means for determining a highest priority one of the plurality of tasks further comprises means for selecting the highest priority one of the plurality of tasks from ones of the plurality of tasks having an associated activity indicator in an active state and wherein the means for determining a next priority one of the plurality of tasks further comprises means for selecting the next priority one of the plurality of tasks from ones of the plurality of tasks having an associated activity indicator in an active state.

23. The system of Claim 22 wherein the means for obtaining an update of the associated one of the plurality of buffer memories with a second portion of the instruction set of the highest priority one of the plurality of tasks when the next instruction for the highest priority one of the plurality of tasks is not in the associated one of the plurality of buffer memories further comprises means for setting the activity indicator for the highest priority one of the plurality of tasks to an inactive state while obtaining an update.

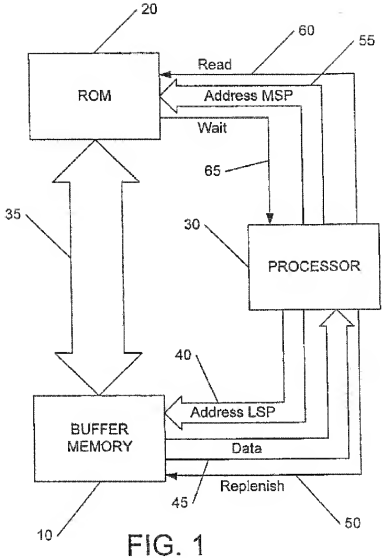
20
60
55
Read
ROM
Address MSP
Wait
65
35
30
PROCESSOR
40
Address LSP
BUFFER
MEMORY
Data
Replenish
45
10
50

FIG. 1

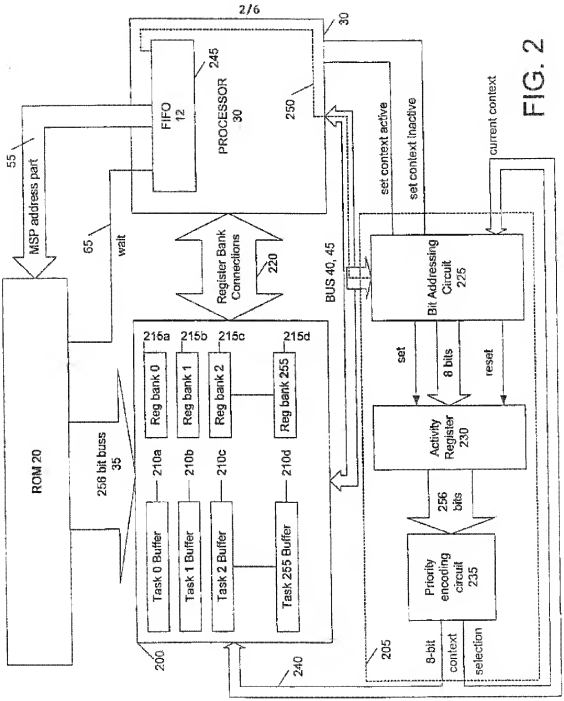
55
MSP address part
65
wait
FIFO
12
PROCESSOR
30
245
250
30
set context active
set context inactive
current context
Register Bank
Connections
220
BUS 40, 45
Bit Addressing
Circuit
225
ROM 20
258 bit buss
35
215a
215b
215c
215d
Reg bank 0
Reg bank 1
Reg bank 2
Reg bank 255
set
8 bits
reset
210a
210b
210c
210d
Task 0 Buffer
Task 1 Buffer
Task 2 Buffer
Task 255 Buffer
Activity
Register
230
256
bits
Priority
encoding
circuit
235
200
240
205
8-bit
context
selection

FIG. 2

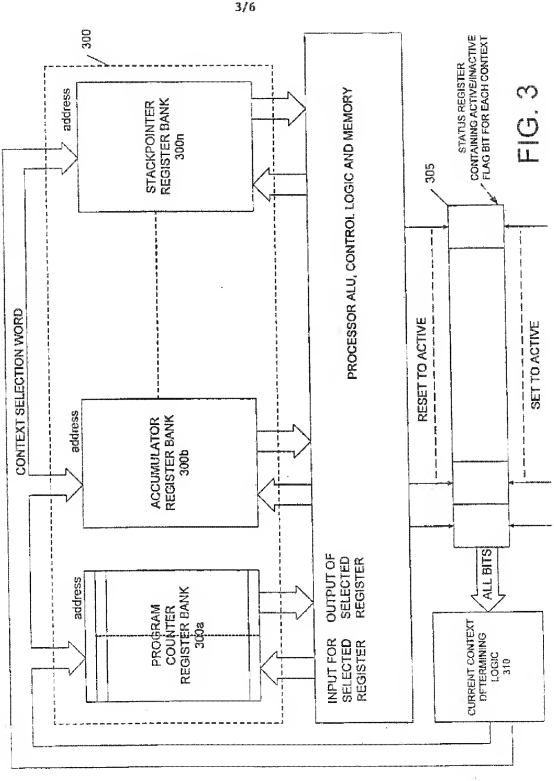
300
CONTEXT SELECTION WORD
address
address
address
PROGRAM COUNTER REGISTER BANK 300a
ACCUMULATOR REGISTER BANK 300b
STACKPOINTER REGISTER BANK 300n
PROCESSOR ALU, CONTROL LOGIC AND MEMORY
INPUT FOR SELECTED REGISTER
OUTPUT OF SELECTED REGISTER
RESET TO ACTIVE
SET TO ACTIVE
305
STATUS REGISTER CONTAINING ACTIVE/INACTIVE FLAG BIT FOR EACH CONTEXT
ALL BITS
CURRENT CONTEXT DETERMINING LOGIC 310

FIG. 3

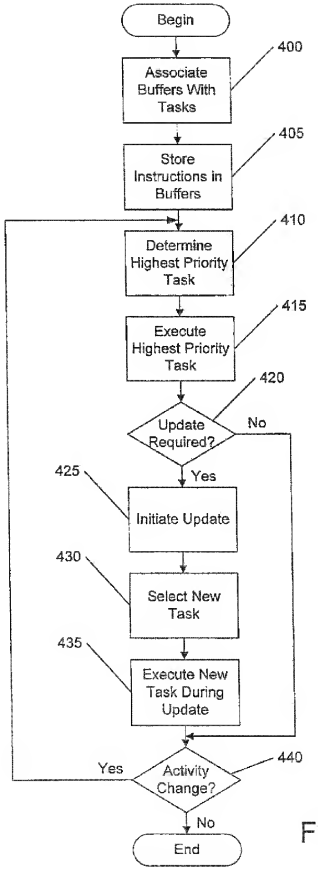
Begin
400
Associate Buffers With Tasks
405
Store Instructions in Buffers
410
Determine Highest Priority Task
415
Execute Highest Priority Task
420
Update Required?
No
Yes
425
Initiate Update
430
Select New Task
435
Execute New Task During Update
440
Activity Change?
Yes
No
End

FIG. 4

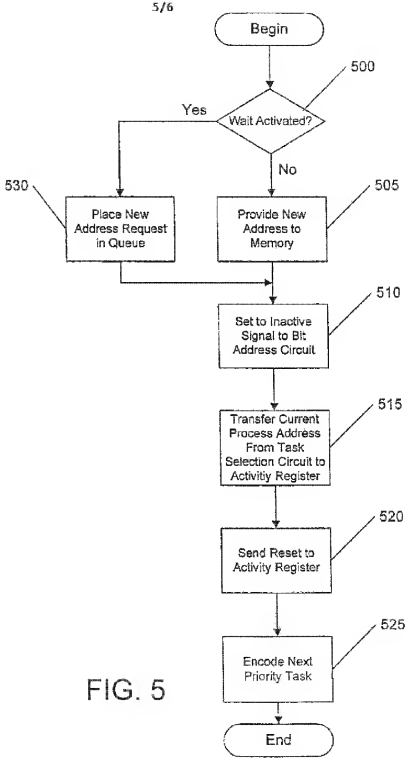
Begin
500
Wait Activated?
Yes
No
530
Place New Address Request in Queue
Provide New Address to Memory
505
510
Set to Inactive Signal to Bit Address Circuit
515
Transfer Current Process Address From Task Selection Circuit to Activitiy Register
520
Send Reset to Activity Register
525
Encode Next Priority Task
End

FIG. 5

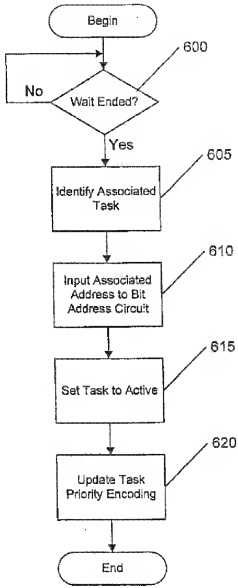
Begin
600
No
Wait Ended?
Yes
605
Identify Associated Task
610
Input Associated Address to Bit Address Circuit
615
Set Task to Active
620
Update Task Priority Encoding
End

FIG. 6

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
31 May 2001 (31.05.2001)

PCT

(10) International Publication Number
WO 01/38970 A3

(51) International Patent Classification[7]: G06F 12/08

(21) International Application Number: PCT/US00/26669

(22) International Filing Date:
28 September 2000 (28.09.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/447,081 22 November 1999 (22.11.1999) US

(71) Applicant: ERICSSON INC [US/US]; 7001 Development Drive, Research Triangle Park, NC 27709 (US).

(72) Inventor: DENT, Paul, W.; 637 Eaglepoint Road, Pittsboro, NC 27312 (US).

(74) Agent: MYERS BIGEL SIBLEY SAJOVEC, P.A.; P.O. Box 37428, Raleigh, NC 27627 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

(88) Date of publication of the international search report:
7 March 2002

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: BUFFER MEMORIES, METHODS AND SYSTEMS FOR BUFFERING HAVING SEPERATE BUFFER MEMORIES FOR EACH OF A PLURALITY OF TASKS

(57) Abstract: Methods, systems and buffer memories are provided that include one buffer per task rather than a single buffer, (the contents of which may be discarded or overwritten when switching tasks in a multi-tasking environment). The rate of cache misses is, therefore, dependent solely on a single task. Furthermore, the present invention provides systems and methods for transferring execution of instructions from one task to another task which may provide reduced overhead and improved utilization of processor cycles concurrently with access operations to a slower ROM to update a buffer memory responsive to a cache miss. The buffer memories are coupled to the ROM by a first, preferably wider, bus and to the processor by a second bus allowing buffer updates to one of the task's buffers while another task is executed from its buffer by the processor. An activity register is coupled to a task selection circuit to identify the highest priority task and select the corresponding buffer as the source of instructions for execution by the processor when the activity state of the various tasks in the multi-tasking environment changes. The activity state of a task is set responsive to the need for instruction execution for the particular task and also by the status of the associated buffer so that a task is set to inactive while buffer updates are performed, thereby allowing a different task to be designated as the highest priority task and executed from its own associated buffer during the buffer update.

WO 01/38970 A3

INTERNATIONAL SEARCH REPORT

Intr tional Application No
PCT/US 00/26669

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F12/08

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal, WPI Data, PAJ, INSPEC, COMPENDEX, IBM-TDB

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 0 856 797 A (SGS THOMSON MICROELECTRONICS) 5 August 1998 (1998-08-05)	1,20
A	the whole document	2-19, 21-23

X	DONGWOOK K ET AL: "A PARTITIONED ON-CHIP VIRTUAL CACHE FOR FAST PROCESSORS" JOURNAL OF SYSTEMS ARCHITECTURE,NL,ELSEVIER SCIENCE PUBLISHERS BV., AMSTERDAM, vol. 43, no. 8, 1 May 1997 (1997-05-01), pages 519-531, XP000685730 ISSN: 1383-7621	1,20
A	the whole document	2-19, 21-23

	-/--	

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
20 July 2001	02/08/2001

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Beltrán-Escavy, J

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Inte ‘tional Application No
PCT/US 00/26669

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication where appropriate, of the relevant passages	Relevant to claim No.
A	EP 0 768 608 A (SUN MICROSYSTEMS INC) 16 April 1997 (1997-04-16) abstract column 1, line 1 -column 4, line 19	1-23
A	EP 0 395 835 A (INTERGRAPH CORP) 7 November 1990 (1990-11-07) abstract column 1, line 1 -column 3, line 26	1-23
A	WO 99 34295 A (MCMZ TECHNOLOGY INNOVATIONS LL) 8 July 1999 (1999-07-08) abstract page 1, line 1 -page 6, line 8	1-23
A	WO 93 09497 A (UNISYS CORP) 13 May 1993 (1993-05-13) abstract page 1, line 1 -page 4, line 6	1-23
A	US 5 442 747 A (CHAN STEVEN S ET AL) 15 August 1995 (1995-08-15) abstract column 1, line 1 -column 2, line 17	1,20
A	US 5 930 821 A (HENRY G GLENN ET AL) 27 July 1999 (1999-07-27) abstract column 1, line 1 -column 4, line 49	1,20
A	US 5 822 757 A (CHI CHI-HUNG) 13 October 1998 (1998-10-13) abstract column 1, line 1 -column 5, line 10	1-20

1

INTERNATIONAL SEARCH REPORT

formation on patent family members

Inte ·tional Application No
PCI/US 00/26669

Patent document cited in search report		Publication date	Patent family member(s)		Publication date
EP 0856797	A	05-08-1998	EP	0856798 A	05-08-1998
			EP	0890149 A	13-01-1999
			WO	9834172 A	06-08-1998
			JP	10232834 A	02-09-1998
			JP	10232839 A	02-09-1998
			JP	11509356 T	17-08-1999
EP 0768608	A	16-04-1997	US	5701432 A	23-12-1997
			JP	9204357 A	05-08-1997
			US	5909695 A	01-06-1999
EP 0395835	A	07-11-1990	CA	2008313 A	03-11-1990
			JP	2302853 A	14-12-1990
WO 9934295	A	08-07-1999	US	6260114 B	10-07-2001
			AU	2204299 A	19-07-1999
			BR	9807274 A	02-05-2000
			CN	1251668 T	26-04-2000
			EP	0972246 A	19-01-2000
WO 9309497	A	13-05-1993	DE	69224649 D	09-04-1998
			DE	69224649 T	25-06-1998
			EP	0611462 A	24-08-1994
			JP	7500936 T	26-01-1995
US 5442747	A	15-08-1995	NONE		
US 5930821	A	27-07-1999	NONE		
US 5822757	A	13-10-1998	DE	69224084 D	26-02-1998
			DE	69224084 T	23-07-1998
			EP	0496439 A	29-07-1992
			JP	4303248 A	27-10-1992